

# Distributed fast boundary element methods

Michal Merta  
IT4Innovations National  
Supercomputing Center  
Ostrava, Czech Republic  
michal.merta@vsb.cz

Jan Zapletal  
IT4Innovations National  
Supercomputing Center  
Ostrava, Czech Republic  
jan.zapletal@vsb.cz

Michal Kravcenko  
IT4Innovations National  
Supercomputing Center  
Ostrava, Czech Republic  
michal.kravcenko@vsb.cz

## ABSTRACT

We present a parallel implementation of the fast boundary element method (BEM) for the Helmholtz equation. After a brief description of BEM, vectorization of the computationally most demanding kernels, and shared memory parallelization, we focus on the distributed memory parallelization using a new approach for distribution of the fast BEM system matrices among computational nodes. Moreover, we present a modification of the adaptive cross approximation (ACA) method capable of dealing with BEM matrices containing zero blocks, which usually lead to problems for the original ACA algorithm. Numerical experiments demonstrating the performance of the implementation were carried out using several Intel architectures.

## KEYWORDS

boundary element method, adaptive cross approximation, parallel computing, SIMD vectorization

### ACM Reference format:

Michal Merta, Jan Zapletal, and Michal Kravcenko. 2018. Distributed fast boundary element methods. In *Proceedings of Supercomputing 2018 conference, Dallas, Texas USA, November 2018 (SC18)*, 3 pages. <https://doi.org/10.1145/nnnnnnn.nnnnnnn>

## 1 INTRODUCTION

The boundary element method (BEM) [8] is well suited for solution of problems stated on unbounded domains (such as electromagnetic or acoustic wave scattering) since it reduces the problem to the boundary of the computational domains. However, the classical BEM produces full matrices and is of quadratic computational complexity with respect to the number of surface DOFs. Moreover, one has to take a special care of the singularities occurring in the numerical integration which leads to a high computational intensity of the integration routines.

An efficient parallel implementation of BEM is necessary to enable solution of large scale real-world problems. The BEM4I library developed at IT4Innovations National Supercomputing Center provides an implementation of BEM parallelized on several levels. On the lowest layer the integration routines are SIMD vectorized using OpenMP pragmas, the system matrix assembly is parallelized in the shared memory by OpenMP threading and the whole procedure can be distributed among computational nodes using MPI. Moreover, to reduce computational complexity and memory requirements one can use a variant of the fast BEM, such as the adaptive cross approximation (ACA) method [1] or the fast multipole method [6].

Here we focus on the distributed memory implementation of the ACA method using novel approach presented in [2, 3] and its application to solution of the sound scattering problems modelled by the Helmholtz equation.

The contribution has the following structure. After a brief introduction to BEM and the BEM4I library, we describe the adaptive cross approximation method and its modification capable of dealing with matrices containing zero blocks. Next, we discuss vectorization and parallelization of the code in the shared memory. The main part focuses on parallelization of fast BEM in the distributed memory. Furthermore, we provide results of numerical experiments demonstrating the efficiency of the implementation.

## 2 INTRA-NODE PARALLELIZATION

Implementation of BEM for the Helmholtz equation has to deal with integrals of the type

$$[V_h]_{\ell,k} := \frac{1}{4\pi} \int_{\tau_\ell} \int_{\tau_k} \frac{e^{i\kappa\|\mathbf{x}-\mathbf{y}\|}}{\|\mathbf{x}-\mathbf{y}\|} d\mathbf{s}_y d\mathbf{s}_x.$$

There are several approaches for treating the singularities occurring in the integrals [7, 8]. However, they result in large computational intensity of the quadrature routines requiring either large number of quadrature nodes or evaluation of expensive transcendental functions. Therefore, the assembly of the system matrices is usually one of the most time-consuming parts of a BEM simulation and we pay significant attention to the parallelization and optimization of the process. On the other hand, its high computational intensity makes BEM capable of utilizing resources of modern multi-/many-core CPUs with wide SIMD registers. Therefore, besides the shared-memory threading by OpenMP, the computationally intensive quadrature routines are vectorized by OpenMP SIMD pragmas, see [4, 5, 9, 10].

## 3 ACA AND ITS MODIFICATION

The ACA algorithm is based on a hierarchical decomposition of the system matrix into blocks and low-rank approximation of admissible blocks using crosses of appropriately chosen rows and columns

$$S_k := \sum_{\ell=1}^k \mathbf{u}_\ell \mathbf{v}_\ell^\top = [\mathbf{u}_1, \mathbf{u}_2, \dots, \mathbf{u}_k][\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_k]^\top = \mathbf{U}_r^{(k)} \mathbf{V}_r^{(k)}.$$

The number of crosses is not given beforehand and the algorithm is stopped when a given accuracy of approximation is achieved. However, when the input matrix has the structure similar to

$$A = \begin{bmatrix} O & A_{1,2} \\ A_{2,1} & O \end{bmatrix},$$

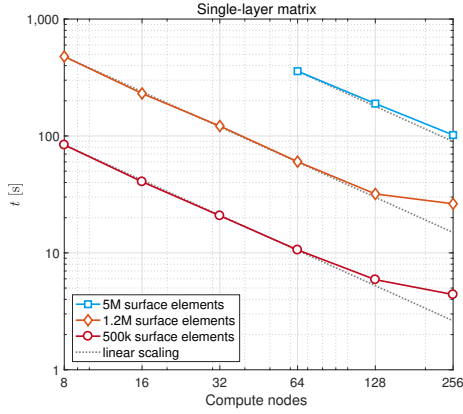


Figure 3.1: Scalability of the system matrix assembly in shared memory.

the algorithm’s partial pivoting only selects nonzero pivots from from either  $A_{1,2}$  or  $A_{2,1}$  ignoring the other block. In the case of BEM, this typically happens when the input geometry contains clusters lying on the same plane. To overcome the problem, in [2] we have presented a modification of the algorithm that identifies problematic blocks in advance, selects references rows and columns from these blocks, and features an outer loop checking the approximation error using the reference rows and columns. Table 3.1 shows a comparison of relative error in  $L_2$  norm for the Helmholtz problem with known solution on a problematic mesh containing large number of clusters lying on the identical plane. Using our approach, we were able to restore a convergence to the exact solution.

#### 4 DISTRIBUTED ACA METHOD

Parallel implementation of the ACA method is not straightforward as one has to deal with load balancing issues. In [3], a new approach for parallelization of ACA was presented. The original boundary mesh is decomposed into  $N$  submeshes, pairs of which define blocks in the system matrix. The related  $N \times N$  submatrices are assigned to  $N$  concurrent processes. Each of the blocks is assembled as a separate low-rank ACA matrix. The distribution minimizing the number of submeshes assigned to a single MPI process is defined using cyclic decompositions of undirected complete graphs. However, in [3] only decompositions for certain unusual number of processes are provided ( $N = 3, 7, 13, 21, \dots$ ). In [2] we extend the

| # surface elements | $L_2$ error          |                      |
|--------------------|----------------------|----------------------|
|                    | Original ACA         | Modified ACA         |
| 10,722             | $2.51 \cdot 10^{-2}$ | $2.51 \cdot 10^{-2}$ |
| 42,888             | $1.35 \cdot 10^{-2}$ | $1.17 \cdot 10^{-2}$ |
| 171,552            | $1.12 \cdot 10^{-2}$ | $5.62 \cdot 10^{-3}$ |
| 686,208            | $4.66 \cdot 10^{-2}$ | $2.75 \cdot 10^{-3}$ |

Table 3.1: Convergence of the original and modified ACA algorithm in  $L_2$  norm.

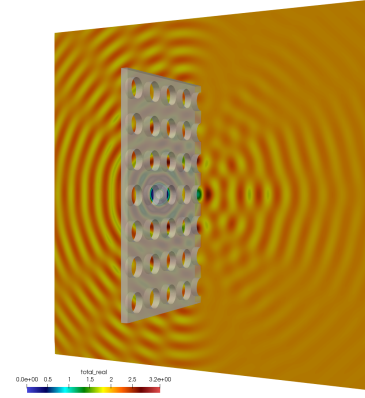


Figure 4.1: Sound scattering off a perforated plate.

approach and provide algorithm for finding decomposition for arbitrary number of processors.

In combination with the above-mentioned shared memory parallelization and vectorization, we obtain a scalable code capable of utilizing resources of modern many- and multi-core clusters. The scalability of the system matrix assembly up to 256 nodes of the Salomon cluster at IT4Innovations (equipped with two 12-core Intel Xeon E5-2680v3 CPUs per node) is presented in Figure 3.1. For the two smaller problems we obtain optimal scalability up to 128 compute nodes; on 256 nodes the local problems are too small to be efficiently parallelized within the shared memory of a single node. The largest problem scales well from 64 to 256 nodes. The computational mesh and solution of the sound scattering problem is depicted in Figure 4.1.

#### 5 CONCLUSION

We have presented a parallel version of the fast boundary element method employing the SIMD vectorization, shared-memory parallelization, and distribution among multiple computational nodes using novel approach based on a cyclic graph decompositions. It is well-suited for problems on unbounded domains, such as modelling of wave propagation. Moreover, the presented version of the ACA method is capable of dealing with usually problematic BEM matrices containing zero blocks.

#### ACKNOWLEDGEMENTS

The work was supported by The Ministry of Education, Youth and Sports from the National Programme of Sustainability (NPS II) project ‘IT4Innovations excellence in science – LQ1602’, from the Large Infrastructures for Research, Experimental Development and Innovations project ‘IT4Innovations National Supercomputing Center – LM2015070’ and by VŠB – Technical University of Ostrava under the grant SP2018/161.

#### REFERENCES

- [1] M. Bebendorf. 2000. Approximation of boundary element matrices. *Numer. Math.* 86 (2000), 565–589.
- [2] M. Kravcenko, M. Merta, and J. Zapletal. [n. d.]. Distributed fast boundary element methods for Helmholtz problems. ([n. d.]). Submitted (2018).

- [3] D. Lukas, P. Kovar, T. Kovarova, and M. Merta. 2015. A parallel fast boundary element method using cyclic graph decompositions. *Numerical Algorithms* 70, 4 (2015), 807–824. <https://doi.org/10.1007/s11075-015-9974-9>
- [4] M. Merta and J. Zapletal. 2015. Acceleration of boundary element method by explicit vectorization. *Advances in Engineering Software* 86 (2015), 70–79. <https://doi.org/10.1016/j.advengsoft.2015.04.008>
- [5] M. Merta, J. Zapletal, and J. Jaros. 2016. Many core acceleration of the boundary element method. *Lecture Notes in Computer Science* 9611 (2016), 116–125. [https://doi.org/10.1007/978-3-319-40361-8\\_8](https://doi.org/10.1007/978-3-319-40361-8_8)
- [6] V. Rokhlin. 1985. Rapid solution of integral equations of classical potential theory. *J. Comput. Phys.* 60, 2 (1985), 187 – 207. [https://doi.org/10.1016/0021-9991\(85\)90002-6](https://doi.org/10.1016/0021-9991(85)90002-6)
- [7] S. Sauter and C. Schwab. 2010. *Boundary Element Methods*. Springer Berlin Heidelberg, Berlin, Heidelberg. [https://doi.org/10.1007/978-3-540-68093-2\\_4](https://doi.org/10.1007/978-3-540-68093-2_4)
- [8] O. Steinbach. 2008. *Numerical Approximation Methods for Elliptic Boundary Value Problems: Finite and Boundary Elements*. Springer, New York.
- [9] J. Zapletal, M. Merta, and L. Maly. 2017. Boundary element quadrature schemes for multi- and many-core architectures. *Computers and Mathematics with Applications* 74, 1 (2017), 157–173. <https://doi.org/10.1016/j.camwa.2017.01.018>
- [10] J. Zapletal, G. Of, and M. Merta. 2017. Parallel and vectorized implementation of analytic evaluation of boundary integral operators. *Computer Methods in Applied Mechanics and Engineering* (2017). Submitted.